BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

LECTURE NOTES

Algorithm and Programming

Minggu 10

Sorting

LEARNING OUTCOMES

LO 3: Evaluasi jenis algoritma, sintaks, dan fungsi untuk pemecahan masalah.

Outcome:

Mahasiswa mampu melakukan evaluasi untuk jenis algoritma, sintaks, dan fungsi dalam pemecahan masalah.

OUTLINE MATERI (Sub-Topic):

- Sorting
- Bubble Sort
- Selection Sort
- Insertion Sort
- Quick Sort
- Merge Sort

ISI MATERI

1. Sorting

Dalam pengolahan data sering kali data tersebut perlu diurutkan (disusun) untuk mempermudah proses data tersebut. Proses pengurutan data tersebut disebut dengan sorting. Salah satu tujuan dari sorting adalah untuk mempercepat pencarian data (searching, retrieving).

Pengurutan data dilakukan berdasarkan nilai kunci (key). Misalnya sejumlah data (record) presetasi akademik mahasiswa yang masing-masing terdiri dari StudentId, nama, dan IPK maka kunci pengurutan dapat berupa StudentId (jika ingin diurutkan berdasarkan StudentId), atau berupa IPK bila ingin didapatkan susunan data dari IPK tertinggi hingga IPK terendah (atau sebaliknya).

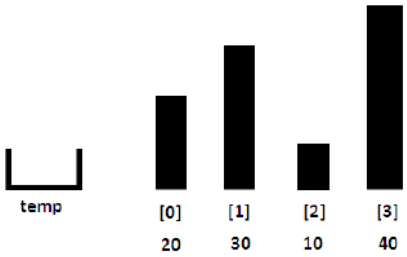
Berdasarkan perbandingan nilai data maka sorting dapat dilakukan dengan 2 cara yatu :

- Ascending: merupakan pengurutan data dari data terkecil hingga data terbesar.
- Descending: merupakan pengurutan data dari data terbesar hingga data terkecil.

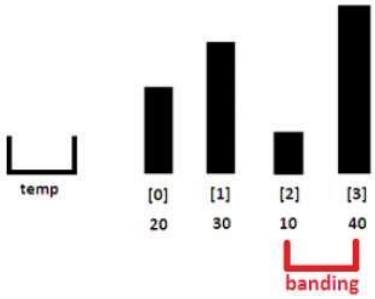
Selanjutnya dalam lecturenote ini akan dijelaskan macam-macam bentuk sorting yang dapat digunakan

2. Bubble Sort

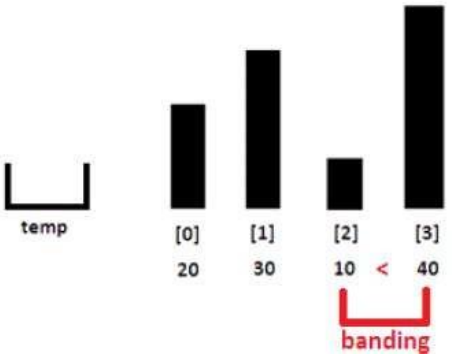
Bubblesort merupakan teknik sorting yang sederhana dan mengikuti prinsip bubble (gelembung udara). Pengurutan dalam algoritma ini menggunakan perbandingan di antara 2 data. Untuk lebih jelasnya perhatikan simulasi berikut ini, simulasi sorting dilakukan secara ascending (pengurutan dari data terkecil hingga data terbesar).



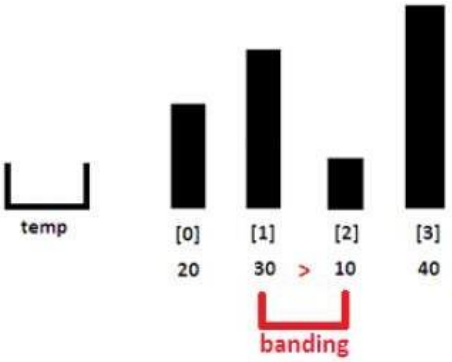
Siapkan variabel bernama "**temp**" Dan kumpulan data yang ingin di sorting dimasukkan ke dalam sebuah variabel berarray sejumlah data yang tersedia



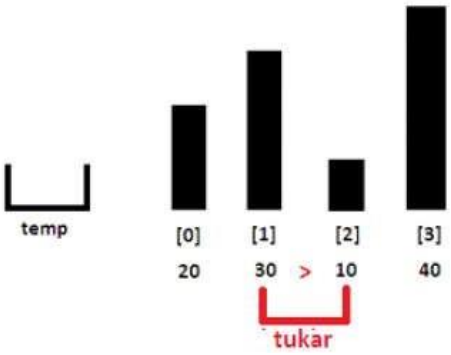
Bandingkan dua data yang terdapat dalam variabel berarray tersebut di mulai dari index array paling belakang



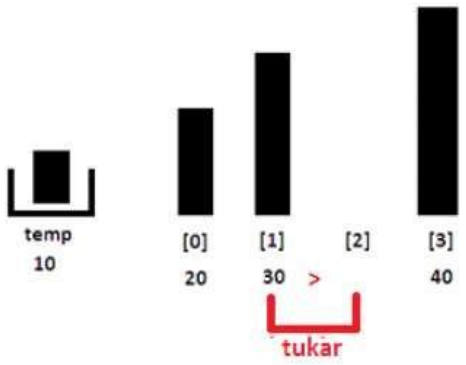
Cek apakah index terakhir lebih besar atau lebih kecil dari index kedua terakhir, jika lebih besar maka tukar kedua data, jika tidak diabaikan, dalam kasus gambar di samping, dua nilai diabaikan (tidak ditukar)



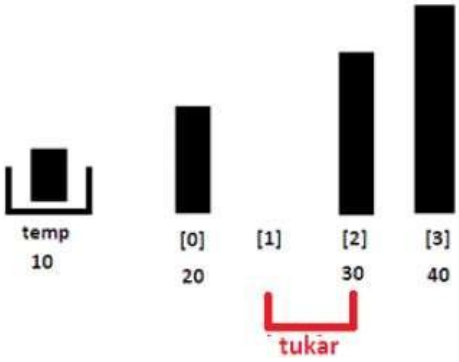
Lanjutkan dengan membandingkan dua nilai selanjutnya.



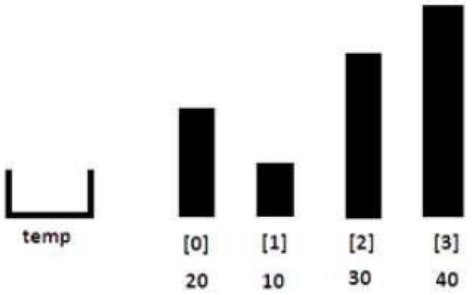
Karena nilai dari index di depannya lebih besar, maka kedua nilai ditukar



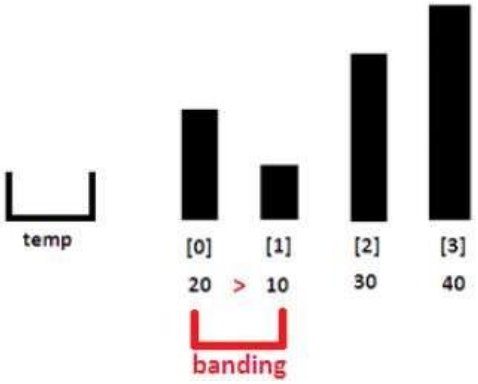
Nilai 10 dari index [2] disimpan terlebih dahulu ke variabel "**temp**" agar nilai tidak hilang pada saat penukaran nilai



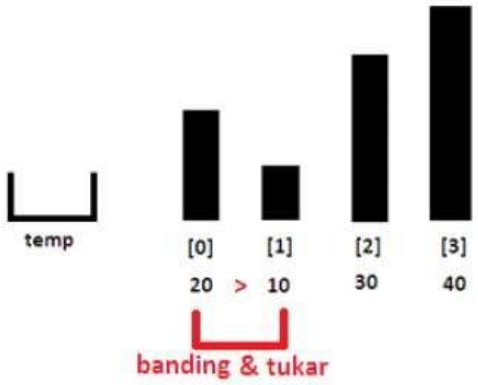
Nilai 30 dipindahkan ke index [2].
Sehingga variabel dengan index [2] bernilai 30 sekarang.



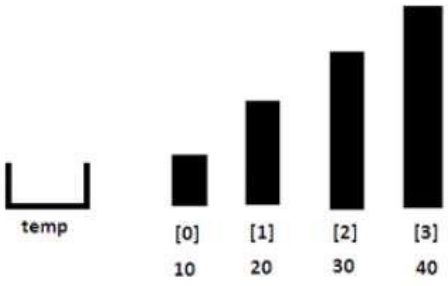
Nilai 10 yang sudah disimpan di temp, kemudian dipindahkan dalam variabel dengan index [1], dalam hal ini penukaran nilai sudah selesai



Lanjutkan dengan membandingkan nilai 2 index selanjutnya



Karena nilai index di depan index [1] lebih besar, maka kedua nilai ditukar dengan langkah yang sama seperti sebelumnya



Cek kembali urutan angka, apakah sudah berurutan secara ascending, bila belum, langkah kembali diulangi dari index terakhir

Berikut merupakan algoritma Bubble sort :

```
void bublesort(int list[], int n)
{
      int i,j;
      for(i=0;i<(n-1);i++)
      for(j=0;j<(n-(i+1));j++)
      if(list[j] > list[j+1])   //dibandingkan
      swap(&list[j],&list[j+1]);
}
```

Berikut merupakan isi dari fungsi swap pada algoritma Bubble sort :

```
void swap(int *x,int *y)
{
       int temp;
       temp = *x;
       *x = *y;
       *y = temp;
}
```

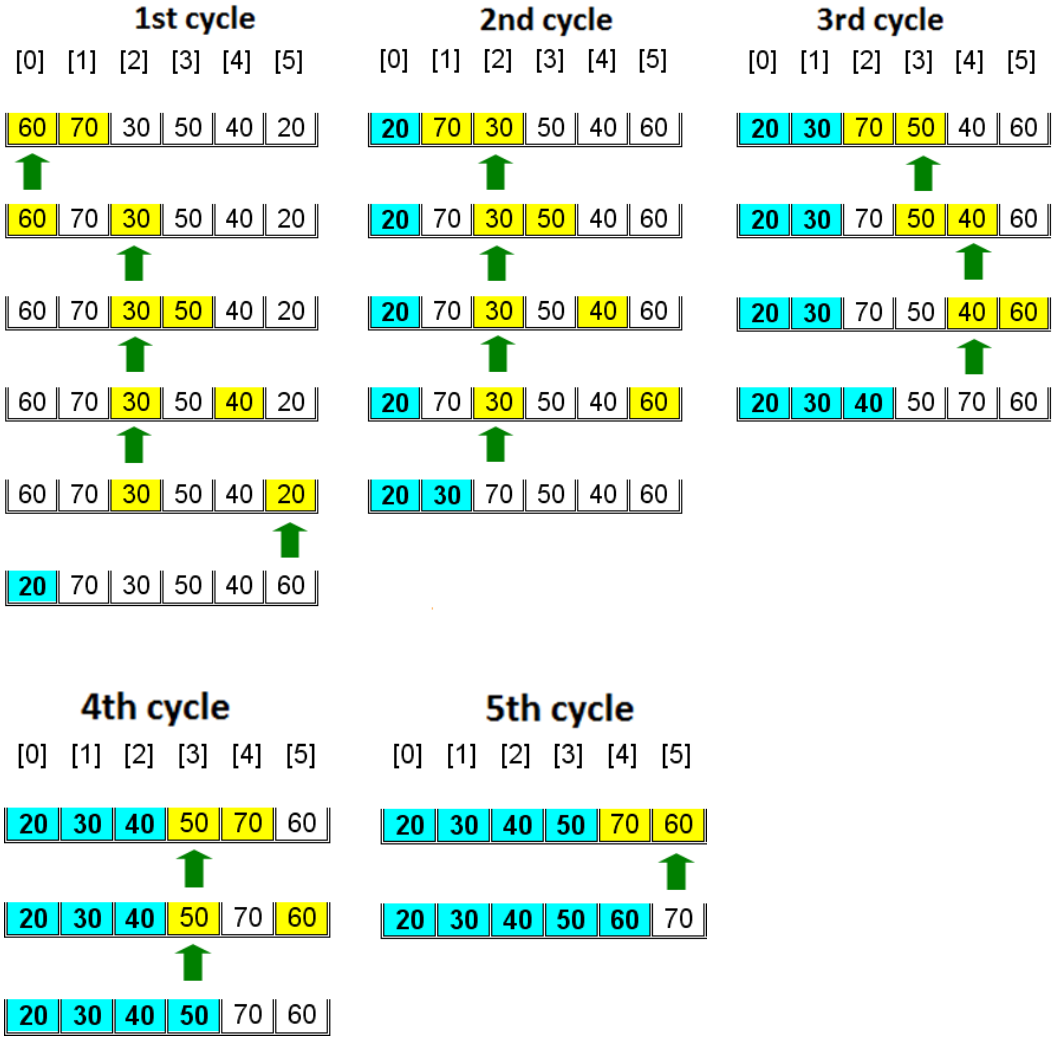
Berikut contoh program Bubble sort pada bahasa C :

```
#include <stdio.h>
#include <stdlib.h>
const int MAX_ELEMENTS = 10;

void swap(int *x,int *y){
      int temp;
      temp = *x;
      *x = *y;
      *y = temp;
}
void bublesort(int list[], int n){
      int i,j;
      for(i=0;i<(n-1);i++)
      for(j=0;j<(n-(i+1));j++)
      if(list[j] > list[j+1])
      swap(&list[j],&list[j+1]);
}

void printlist(int list[],int n){
      int i;
      for(i=0;i<n;i++)
      printf("%d\t",list[i]);
}

int main(){
      int list[MAX_ELEMENTS];
      int i = 0;
      // generate random numbers and fill them to the list
      for(i = 0; i < MAX_ELEMENTS; i++ ){
      list[i] = rand();
      }
      printf("The list before sorting is:\n");
      printlist(list,MAX_ELEMENTS);

      // sort the list
      bublesort(list,MAX_ELEMENTS);

      // print the result
      printf("The list after sorting using bubble sorting
      algorithm:\n");
      printlist(list,MAX_ELEMENTS);
      return 0;
}
```

3. Selection Sort

Algoritma selection sort membagi proses pengurutan menjadi putaran-putaran. Pada putaran pertama diseleksi data dengan nilai terkecil dan data ini ditempatkan pada posisi indeks terkecil (data[0]). Pada putaran ke-2 diseleksi data dengan nilai nomor dua kecil untuk ditempatkan pada posisi ke-2 (data[1]). Proses seleksi dalam satu putaran dilakukan dengan membandingkan nilai elemen pada 2 indeks pembanding.

Selama proses pembanding berlangsung perubahan hanya dilakukan terhadap indeks pembanding. Pertukaran data secara fisik dilakukan pada akhir setiap putaran. Jadi, berbeda dengan bubble sort yang pada setiap pembandingan dilakukan pertukaran data bila diharuskan, selection sort melakukan pertukaran data hanya satu kali untuk setiap putaran yaitu saat putaran tersebut selesai. Berikut simulasi dari selection sort yang dilakukan secara ascending (pengurutan dari data terkecil hingga data terbesar).



Berikut merupakan algoritma Selection sort :

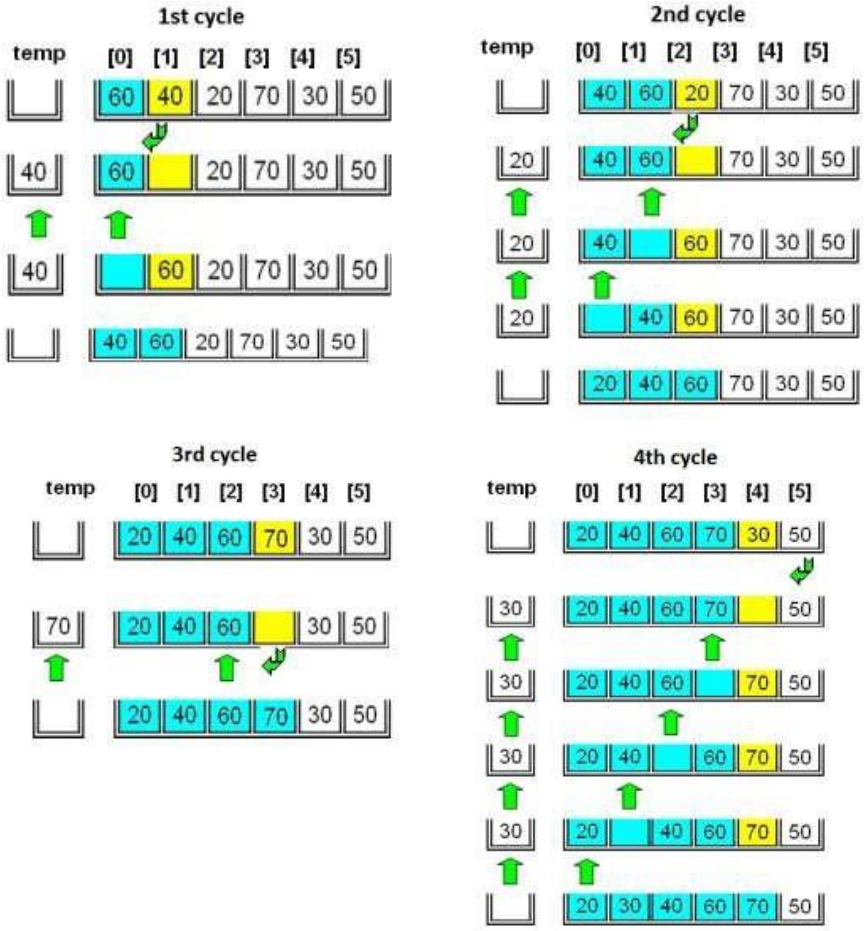
```
for(i=0; i<=N-2; i++){          /* N=number of data */
     for(j=i; j<=N-1; j++){
          Note the index of smallest value between A[j] s/d A[N-1],
          Save it in variable k.
          Swap A[i] with A[k].
     }
}
```

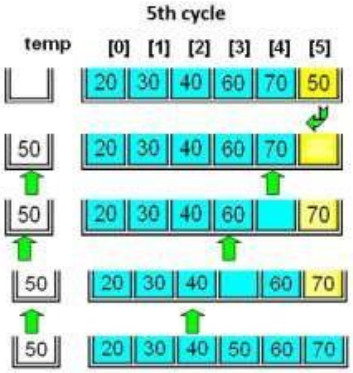
Berikut contoh program Selection sort pada bahasa C :

```c
#include <stdio.h>

void tukar(int *n1,int *n2){
    int n3;
    n3=*n1;
    *n1=*n2;
    *n2=n3;
}
void selectionsort(int bil[],int banyak){
    int i,j,posisi;

    for(i=0;i<banyak-1;i++){
        posisi=i;
        for(j=i;j<banyak;j++){
            if(bil[j]<bil[posisi])
                posisi=j;
        }
        if(posisi!=i)
            tukar(&bil[posisi],&bil[i]);
    }

    printf("\nHasil setelah sort : \n");
    for(i=0;i<banyak;i++){
        printf("%d ",bil[i]);
    }
}

int main(){
    int banyak,i,bil[10];

    do{
        printf("Masukkan banyaknya bilangan [1-10] ? ");
        scanf("%d",&banyak); fflush(stdin);
    }
    while(banyak<1||banyak>10);
    for(i=0;i<banyak;i++){
        printf("Masukkan bilangan ke-%d : ",i+1);
        scanf("%d",&bil[i]); fflush(stdin);
    }
    selectionsort(bil,banyak);
    getchar();
    return 0;
}
```

4. Insertion Sort

Algoritma insertion sort prosesnya sama seperti seseorang mengurutkan kartu. Selembar demi selembar kartu diambil dari kumpulan kartu dan disisipakan pada posisi yang tepat. Pada putaran pertama diurutkan 2 data pertama. Perngurutan ini sifatnya relatif, artinya kedua data ini belum tentu memiliki dua data terkecil dari seluruh data melainkan bahwa kedua data ini telah terurut dalam lingkup dua kartu tersebut. Pada putaran kedua dicarikan tempat sisip yang tepat bagi data[2].Berikut simulasi dari insertion sort yang dilakukan secara ascending (pengurutan dari data terkecil hingga data terbesar) :





Berikut merupakan algoritma Selection sort :

```
void Insertion(int *Arr, int n){
      int i, k, y;
      for(k=1; k < n; k++) {
         y = Arr[k];
         for(i=k-1; i >= 0 && y < Arr[i]; i--)
            Arr[i+1] = Arr[i];
         Arr[i+1] = y;
         }
}
```

Berikut contoh program Insertion sort pada bahasa C :

```
#include <stdio.h>

void insertionsort(int bil[],int banyak){
    int i,j,temp;

    for(i=1;i<banyak;i++){
        temp=bil[i];
        for(j=i-1;j>=0 && bil[j]>temp;j--){
            bil[j+1]=bil[j];
        }
        bil[j+1]=temp;
    }
    printf("\nHasil setelah sort : \n");
    for(i=0;i<banyak;i++){
        printf("%d ",bil[i]);
    }
}
```

```
int main(){
    int banyak,i,bil[10];

    do{
        printf("Masukkan banyaknya bilangan [1-10] ? ");
        scanf("%d",&banyak); fflush(stdin);
    }
    while(banyak<1||banyak>10);
    for(i=0;i<banyak;i++){
        printf("Masukkan bilangan ke-%d : ",i+1);
        scanf("%d",&bil[i]);  fflush(stdin);
    }
    insertionsort(bil,banyak);

    getchar();
    return 0;
}
```

5. Quick Sort

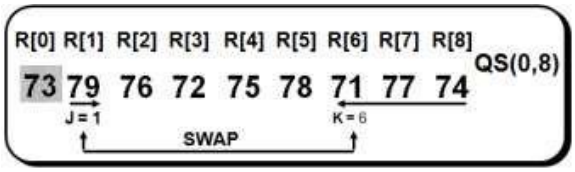
Algoritma quick sort bekerja dengan membagi sekumpulan data menjadi 2 bagian sedemikian rupa sehingga elemen tertentu (elemen ke-i) berada tepat pada posisinya, semua elemen yang nilainya lebih kecil dari elemen ke-i berada pada di sebelah kiri elemen ke-i dan semua elemen yang nilainya lebih besar dari elemen ke-i berada di sebelah kanan elemen ke-i.

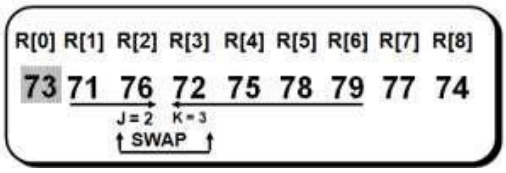
a[kiri], a[kiri+1], ... , a[i-1] < a[i] < a[i+1], a[i+2], ..., a[kanan]

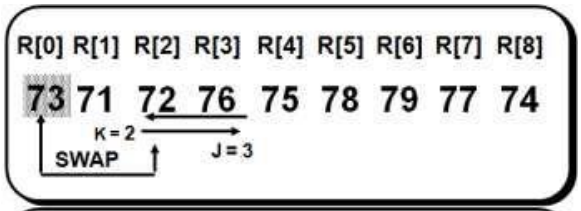
Selanjutnya elemen-elemen pada posisi sebelah kiri elemen ke-i di-sort kembali dengan cara yang sama, sama halnya dengan elemen-elemen pada posisi sebelah kanan elemen ke-i.

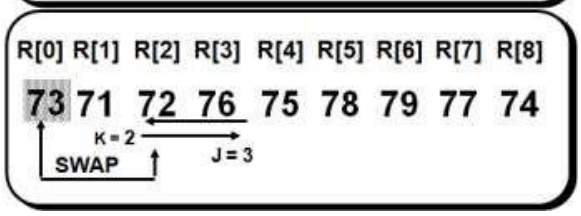
Langkah untuk menentukan poisis yang sebenarnya bagi elemen paling kiri (data[kiri]) sesuai dengan algoritma dari quick sort adalah sebagai berikut :

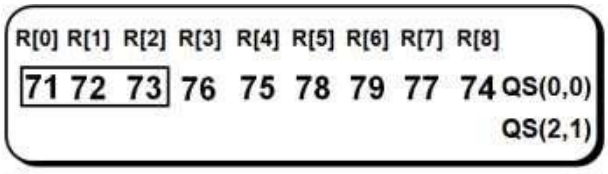
- Cari data arah kiri ke kanan data yang nilainya tidak lebih kecil dari data[kiri], misalnya data tersebut data[j] (instruksi baris ke-7).
- Cari data arah kanan ke kiri data yang nilainya tidak lebih besar daripada data[kiri], misalnya data tersebut data[k] (instruksi baris ke-8).
- Apanila j < k maka tukar data[j] dan data[k] (instruksi baris ke-9) dan ulangi kembali langkah pertama sampai dengan ketiga (instruksi baris ke-10).
- Tukar data[kiri] dan data[k], data[k] telah berada pada posisinya yang tepat (instruksi baris ke-11).
- Ulangi langkah pertama samapi dengan keempat untuk data sebelah kiri (data[kiri] sampai data[k-1]) dan data sebelah kanan (data[k+1] sampai dengan data[kanan]), instruksi baris ke-12 dan ke-13. Proses pengulangan ini berlangsung selama kiri < kanan.

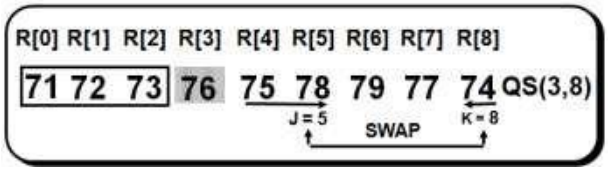

R[0] R[1] R[2] R[3] R[4] R[5] R[6] R[7] R[8]
73 79 76 72 75 78 71 77 74
QS(0,8)
J = 1
K = 6
SWAP

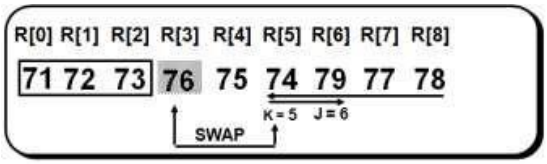

R[0] R[1] R[2] R[3] R[4] R[5] R[6] R[7] R[8]
73 71 76 72 75 78 79 77 74
J = 2
K = 3
SWAP

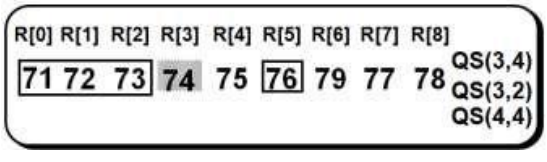

R[0] R[1] R[2] R[3] R[4] R[5] R[6] R[7] R[8]
73 71 72 76 75 78 79 77 74
K = 2
J = 3
SWAP

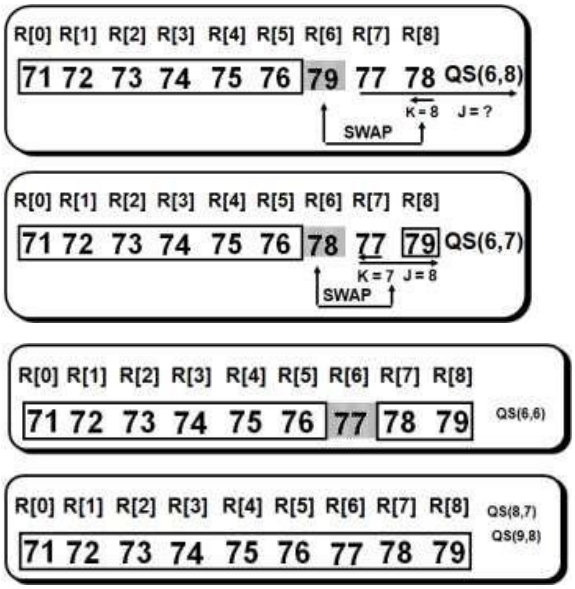

R[0] R[1] R[2] R[3] R[4] R[5] R[6] R[7] R[8]
73 71 72 76 75 78 79 77 74
K = 2
J = 3
SWAP


R[0] R[1] R[2] R[3] R[4] R[5] R[6] R[7] R[8]
71 72 73 76 75 78 79 77 74
QS(0,0)
QS(2,1)


R[0] R[1] R[2] R[3] R[4] R[5] R[6] R[7] R[8]
71 72 73 76 75 78 79 77 74
QS(3,8)
J = 5
K = 8
SWAP


R[0] R[1] R[2] R[3] R[4] R[5] R[6] R[7] R[8]
71 72 73 76 75 74 79 77 78
K = 5
J = 6
SWAP


R[0] R[1] R[2] R[3] R[4] R[5] R[6] R[7] R[8]
71 72 73 74 75 76 79 77 78
QS(3,4)
QS(3,2)
QS(4,4)



Berikut algoritma dari quick sort :

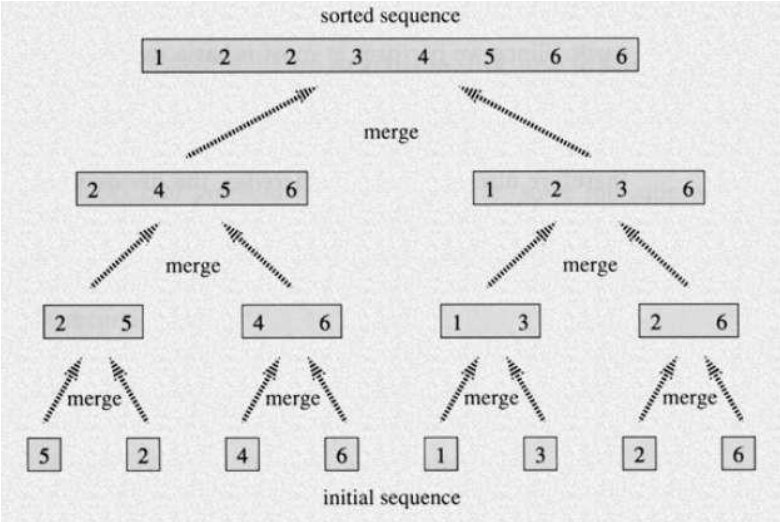
```
/*1*/ void quicksort(int *bil,int kr,int kn){
/*2*/    int i,j,k;
/*3*/    if(kr<kn){
/*4*/        j=kr;
/*5*/        k=kn+1;
/*6*/        do{
/*7*/            do j++; while(j<=kn && bil[j] < bil[kr]);
/*8*/            do k--; while(bil[k] > bil[kr]);
/*9*/        if(j<k) tukar(&bil[j],&bil[k]);
/*10*/       }while(j<=k);
/*11*/       tukar(&bil[kr],&bil[k]);
/*12*/       quicksort(bil,kr,k-1);
/*13*/       quicksort(bil,k+1,kn);
/*14*/    }
/*15*/ }
```

Berikut contoh program dari quick sort :

```
#include <stdio.h>

void tukar(int *n1,int *n2){
    int n3;
    n3=*n1;
    *n1=*n2;
    *n2=n3;
}

void quicksort(int *bil,int kr,int kn){
    int i,j,k;
    if(kr<kn){
        j=kr;
        k=kn+1;
        do{
            do j++; while(j<=kn && bil[j] < bil[kr]);
            do k--; while(bil[k] > bil[kr]);
            if(j<k)
                tukar(&bil[j],&bil[k]);
        }while(j<=k);
        tukar(&bil[kr],&bil[k]);
        quicksort(bil,kr,k-1);
        quicksort(bil,k+1,kn);
    }
}
void cetak(int bil[],int banyak){
    int i;
    printf("\nHasil setelah sort : \n");
    for(i=0;i<banyak;i++){
        printf("%d ",bil[i]);
    }
}

int main(){
    int banyak,i,bil[10];

    do{
        printf("Masukkan banyaknya bilangan [1-10] ? ");
        scanf("%d",&banyak); fflush(stdin);
    }while(banyak<1||banyak>10);
    for(i=0;i<banyak;i++){
        printf("Masukkan bilangan ke-%d : ",i+1);
        scanf("%d",&bil[i]); fflush(stdin);
    }
    quicksort(bil,0,banyak-1);
    cetak(bil,banyak);
    getchar();
    return 0;
}
```

6. Merge Sort

Algoritma merge sort adalah pengurutan dengan cara penggabungan. Dua kumpulan data yang masing-masing telah diurutkan digabung menjadi satu. Penggabungan dimulai dengan menggabungkan kelompok data dengan jumlah elemen terkecil, yaitu kelompok satu data digabung dengan kelompok satu data. Berikut simulasi dari algoritma merge sort :



.

SIMPULAN

- Dalam pengolahan data sering kali data tersebut perlu diurutkan (disusun) untuk mempermudah proses data tersebut. Proses pengurutan data tersebut disebut dengan sorting. Salah satu tujuan dari sorting adalah untuk mempercepat pencarian data (searching, retrieving).
- Berdasarkan perbandingan nilai data maka sorting dapat dilakukan dengan 2 cara yatu ascending dan descending.
- Bubblesort merupakan teknik sorting yang sederhana dan mengikuti prinsip bubble (gelembung udara). Pengurutan dalam algoritma ini menggunakan perbandingan di antara 2 data.
- Algoritma selection sort membagi proses pengurutan menjadi putaran-putaran. Pada putaran pertama diseleksi data dengan nilai terkecil dan data ini ditempatkan pada posisi indeks terkecil (data[0]).
- Algoritma insertion sort prosesnya sama seperti seseorang mengurutkan kartu. Selembar demi selembar kartu diambil dari kumpulan kartu dan disisipakan pada posisi yang tepat.
- Algoritma quick sort bekerja dengan membagi sekumpulan data menjadi 2 bagian sedemikian rupa sehingga elemen tertentu (elemen ke-i) berada tepat pada posisinya, semua elemen yang nilainya lebih kecil dari elemen ke-i berada pada di sebelah kiri elemen ke-i dan semua elemen yang nilainya lebih besar dari elemen ke-i berada di sebelah kanan elemen ke-i.
- Algoritma merge sort adalah pengurutan dengan cara penggabungan. Dua kumpulan data yang masing-masing telah diurutkan digabung menjadi satu. Penggabungan dimulai dengan menggabungkan kelompok data dengan jumlah elemen terkecil.

DAFTAR PUSTAKA

- Paul J. Deitel & Harvey. Deitel. (2022). C how to program. 09. Pearson Education. Hoboken. ISBN:978-0-13-739839-3. Chapter 13
- Sorting Algorithm Animations: http://www.sorting-algorithms.com/
- Sorting Algorithms: http://www.youtube.com/watch?v=INHF_5RIxTE